Auxiliary Learning Induced Graph Convolutional Networks

Anonymous Author(s)
Affiliation
Address
email

A Appendix

A.1 Compared Methods

GCN: The vanilla Graph Convolutional Network (GCN) [1]. The proposed method would collapse into a three-layer GCN model ff removing the label generator and the link predictor from it. For comparison, we use a three-layer GCN (GCN3) and a two-layer GCN (GCN2) both with hidden layers size of 64 units.

GAT: Graph Attention Networks (GAT) [2] is a spacial-based graph neural networks, which involves masked self-attentional layers. The compared GAT method contains 2 graph attentional layers, where the first layer consists of 8 attention heads and the second layer consists of a single attention head that computes C features as the classifier, where C is the number of classes.

DualGCN: Dual Graph Convolutional Networks (DualGCN) [3] is a GCN-based graph neural networks. It contains 2 GCN networks in parallel with hidden layer size of 64 units. Dropout rate is set to 0.1, and windows size is set to 2, which follows the parameters setting in [3].

SGC: Simple Graph Convolution [4] (SGC) is a simplified version of GCN. Following gfNN[5], we use two-layer Feedforward network as a non-linear classification in our implementation.

APPNP: Approximation of the Personalized Propagation of Neural Predictions [6] is a spacial-based graph neural network. We use two layers with 64 hidden unit and 10 power iteration steps as the author recommended.

AL-GCN (ours): It is the proposed method. We use a three-layer GCN with the hidden layers size of 64 units and 64 units as the backbone network. The first two layers are considered as a feature extractor h_{θ_2} and the last layer is considered as a classifier f_{θ_1}. A label generator is a two-layer GCN with hidden layer size of 64 units, and a link predictor is a decoder. Our method has two variants. Al-GCN2 and AL-GCN3, which have two GCN layers and three GCN layers, respectively. By default, AL-GCN has three GCN layers, which corresponds to AL-GCN3.

A.2 More experimental results

A.2.1 On the training of the backbone networks

The proposed method uses the pseudo soft labels generated by the label generator to update the node classifier instead of the entire backbone networks. We conduct the following ablation experiment: the generated pseudo soft labels are applied to update the entire backbone networks instead of only the node classifier, termed GCN-w. The comparison is listed in Table 1. Compared to the proposed method, the learning performance of Al-GCN-w is degraded slightly. The experimental results demonstrate that the training strategy is effective via updating a node classifier instead of the entire backbone networks.

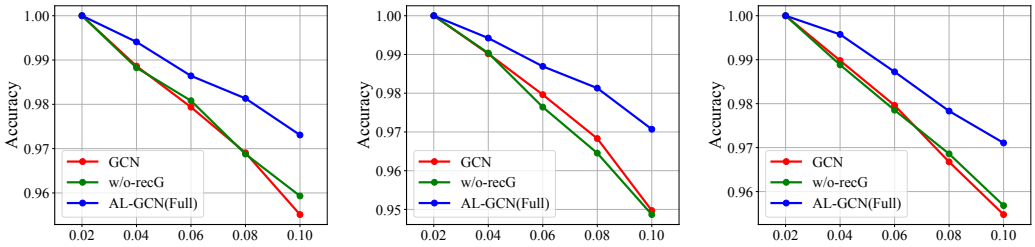
Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Table 1: Comparison of the updating of GCN layer.

Method	Cora	Citeseer	Pubmed
GCN	$80.7 \pm 1.2\%$	$68.0 \pm 1.4\%$	$77.7 \pm 0.5\%$
Al-GCN-w	$83.1 \pm 0.6\%$	$71.7 \pm 0.6\%$	$80.4 \pm 0.5\%$
AL-GCN	$84.7 \pm 0.4\%$	$72.3 \pm 0.5\%$	$81.4 \pm 0.6\%$

Table 2: Comparison of the auxiliary and meta auxiliary training

Method	Cora	Citeseer	Pubmed
GCN	$80.7 \pm 1.2\%$	$68.0 \pm 1.4\%$	$77.7 \pm 0.5\%$
non-meta-P	$83.2 \pm 0.6\%$	$71.5 \pm 0.8\%$	$80.9 \pm 0.4\%$
non-meta-G	$84.5 \pm 0.5\%$	$71.9 \pm 0.5\%$	$81.2 \pm 0.5\%$
non-meta-M	$83.3 \pm 0.6\%$	$71.6 \pm 0.7\%$	$80.8 \pm 0.5\%$
Al-GCN	$84.7 \pm 0.4\%$	$72.3 \pm 0.5\%$	$81.4 \pm 0.6\%$



(a) Cora (b) Citeseer (c) Pubmed

Figure 1: The relative classification accuracy vs noise rate nr.

A.2.2 On the meta auxiliary learning scheme

Different from the common training approach, the meta auxiliary learning strategy concerns more about the performance of the primary task. To know how meta-learning affects the node classification performance, we design the different training strategy instead of the meta training strategy and compare it to the proposed method: (1) A link predictor is trained commonly instead of meta training, termed non-meta-P, (2) A label generator is trained commonly instead of meta training, termed non-meta-G, and (3) Both link predictor and label generator are trained commonly without meta training, termed non-meta-M. The experimental results are listed in Table 2.

As shown in Table 2, compared with the common training strategy, the meta auxiliary learning strategy can significantly improve the auxiliary effect of the primary task. The proposed method is superior to the three variants of our method. Specifically, our method significantly and consistently outperforms non-meta-M, which demonstrates that the meta auxiliary learning approach can more effectively train the primary node classifier.

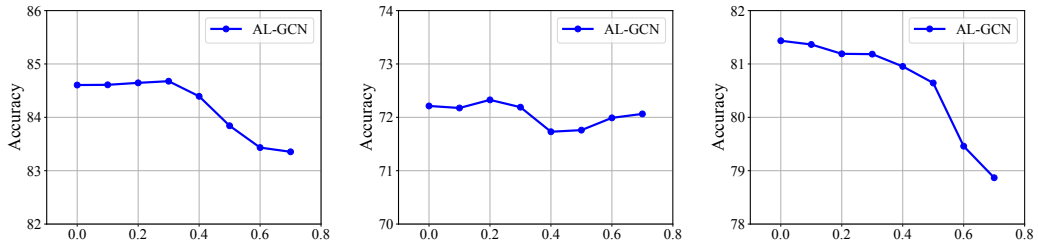
The reason is attributed that The common training strategy trains each task independently. However, the meta auxiliary learning strategy will further consider the performance of the primary task after the parameters are updated according to the auxiliary task. Thus, the learning performance is enhanced.

A.2.3 Noisy Edge

As we use the reconstructed graph as input of the feature extractor, our model is not sensitive to a small amount of noisy edge. In this experiment, we randomly replace a set of real edges E to not existing fake edges with a specific rate nr, and the result is shown in Fig. 1. In order to show the robustness of the model to noise more clearly, we use the training result of introducing 2% noise as the baseline to calculate the relative performance of the model after introducing more noise.

Table 3: The existence of edges in the citation dataset

Dataset		1-hop	2-hop	3-hop	4-hop
Cora	aveP	100%	6.18%	3.21%	0.97%
	Edges	5,278	46,010	164,572	494,367
Citeseer	aveP	100%	8.95%	6.92%	2.14%
	Edges	4,552	20,771	65,799	151,503
Pubmed	aveP	100%	2.75%	1.21%	0.19%
	Edges	44,324	553,034	3,676,040	19,258,260



(a) Cora (b) Citeseer (c) Pubmed

Figure 2: The classification accuracy vs parameter τ.

A.2.4 On speeding up model training

In practice, the edges in a graph are sparse. It is unnecessary to predict the edge existence probability between every node pairs when calculating the reconstructed graph adjacency matrix. Thus, it is vital to choose a subset of neighbors to speed up the implementation of the algorithm. To avoid large computational burden in the implementation of the algorithm, we adopt a graph sampling strategy to reduce computational complexity.

Denoting the k-hop neighbors of node v_i with $\mathcal{N}^{(k)}(v_i)$, we calculate the average probability as,

$$\text{ave}P^{(k)} = \frac{1}{N}\sum_{v_i \in \mathcal{V}} P(v_j \in \mathcal{N}^{(1)}(v_i) | v_j \in \mathcal{N}^{(k)}(v_i)), \tag{1}$$

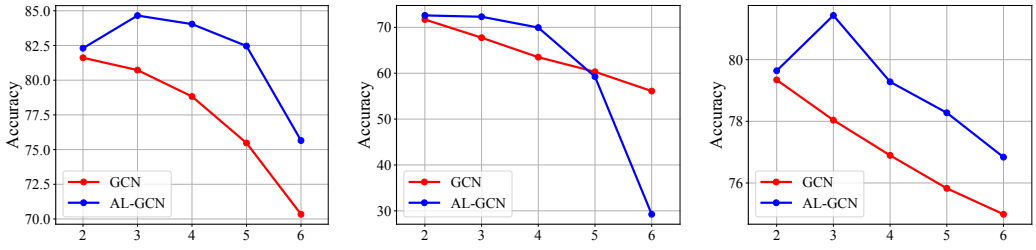
where $\text{ave}P^{(k)}$ means that the average probability for all nodes whose k-hop neighbors are also their 1-hop neighbors, i.e., there is an edge between node v_i and node $v_j \in \mathcal{N}^{(k)}(v_i)$. As shown in Table 3, the existence probability of edge (v_i, v_j) decreases as k increases, where $v_j \in \mathcal{N}^{(k)}(v_i)$ is a k-hop neighbor of v_i.

According to the results, it can be inferred that most of the potential edges are concentrated in the nodes and their 2-hop neighbors. To sufficiently compute the reconstructed graph adjacency matrix, we firstly build a 2-hop graph which assumes that every node and their 2-hop neighbors are connected, where the edge set of the 2-hop graph is defined as $\mathcal{E}_{2-\text{hop}} = \{e_{ij} | v_j \in \mathcal{N}^{(2)}(v_j)\}$. In each training iteration, we select edges from the reconstructed graph adjacency matrix in the last iteration with edge weight larger than threshold τ, as $\mathcal{E}'_{\text{recon}} = \{\mathbf{e} | \mathbf{e} \in \mathcal{E}^{(t-1)}_{\text{recon}}, |\mathbf{e}| > \tau\}$. And then, we sample a set of edges from the 2-hop graph as $\mathcal{E}'_{2-\text{hop}} \subset \mathcal{E}_{2-\text{hop}}$ to expand the reconstructed graph adjacency matrix and the final reconstructed graph adjacency matrix used in the t^{th} training iteration becomes $\mathcal{E}^{(t)}_{\text{recon}} = \mathcal{E}'_{\text{recon}} \bigcup \mathcal{E}'_{2-\text{hop}}$. Finally, the weights of edges in $\mathcal{E}^{(t)}_{\text{recon}}$ are predicted by the decoder.

To observe how threshold τ affects the model perforamnce, we conduct the experiments by varying the different numbers of threshold τ. The results is shown in Fig. 2

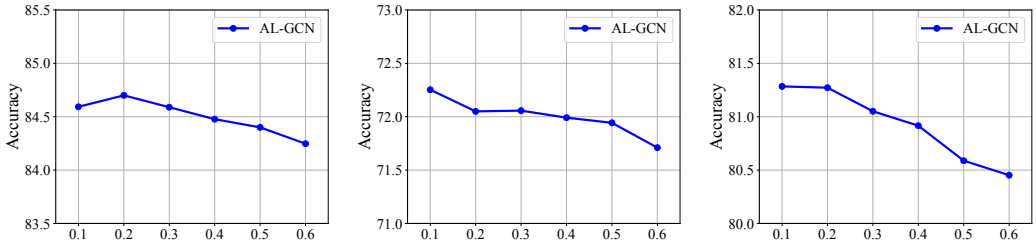
A.2.5 Parameter Study

The proposed method contains two key hyper-parameter, which affects the classification results.



(a) Cora (b) Citeseer (c) Pubmed

Figure 3: The classification accuracy vs parameter K.



(a) Cora (b) Citeseer (c) Pubmed

Figure 4: The classification accuracy vs parameter r.

1. *Model layers* K. The message propagation mechanism of GCN is equivalent to applying Laplacian smoothing to the graph to eliminate high-frequency noise. However, stacking too much layers will cause the over-smoothing problem and performance degradation.
2. *Sampling rate* r of the training edge set for link prediction. We randomly sample the certain percentage of the edges as the input graph for message passing, and predict the existence of edges between each nodes based on the derived hidden embedding. Let $\mathcal{E}_{\text{train}}$ denote the edge sets used in training, and $\mathcal{E}$ denote the set of all edges in a graph, then the sampling rate r is defined as $r = |\mathcal{E}_{\text{train}}|/|\mathcal{E}|$.

We conduct the experiments by varying the different numbers of layer K or the different sampling rates. The experimental results are shown in Fig. 3 and 4.

References

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017.

[2] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations*, 2018.

[3] Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 499–508, 2018.

[4] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6861–6871, 2019.

[5] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *CoRR*, abs/1905.09550, 2019.

[6] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations*, 2019.